

[illegible]

```

LL          IIIIII      SSSSSSSS
LL          IIIIII      SSSSSSSS
LL          II         SS
LL          II         SS
LL          II         SS
LL          II         SS
LL          II         SSSSSS
LL          II         SSSSSS
LL          II         SS
LL          II         SS
LL          II         SS
LL          II         SS
LLLLLLLLLLL IIIIIIII   SSSSSSSS
LLLLLLLLLLL IIIIIIII   SSSSSSSS

```



```
1 0001 0 MODULE showsystem (IDENT = 'V04-000',
2 0002 0 ADDRESSING_MODE (EXTERNAL = GENERAL)) =
3 0003 0
4 0004 1 BEGIN
5 0005 1
6 0006 1
7 0007 1 *****
8 0008 1 *
9 0009 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY *
10 0010 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS. *
11 0011 1 * ALL RIGHTS RESERVED. *
12 0012 1 *
13 0013 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED *
14 0014 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE *
15 0015 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER *
16 0016 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY *
17 0017 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY *
18 0018 1 * TRANSFERRED. *
19 0019 1 *
20 0020 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE *
21 0021 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT *
22 0022 1 * CORPORATION. *
23 0023 1 *
24 0024 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS *
25 0025 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL. *
26 0026 1 *
27 0027 1 *
28 0028 1 *****
29 0029 1
30 0030 1
31 0031 1 ++
32 0032 1
33 0033 1 FACILITY: SHOW utility
34 0034 1
35 0035 1 ABSTRACT:
36 0036 1 This module contains the routines for the SHOW SYSTEM command
37 0037 1
38 0038 1 ENVIRONMENT:
39 0039 1 VAX native, user and kernel mode
40 0040 1
41 0041 1 AUTHOR: Gerry Smith CREATION DATE: 30-Jul-1982
42 0042 1
43 0043 1 MODIFIED BY:
44 0044 1
45 0045 1 V03-007 AEW0002 Anne E. Warner 27-Feb-1984
46 0046 1 Reorganize 'Ph.Mem' format to handle increased process
47 0047 1 working set sizes.
48 0048 1
49 0049 1 V03-006 AEW0001 Anne E. Warner 02-Feb-1984
50 0050 1 Reorganize the SHOW SYSTEM display.
51 0051 1 - Make the display fit on an 80 character display by
52 0052 1 taking out the UIC.
53 0053 1 - Add the qualifier /FULL to display all information
54 0054 1 plus add a second line with the UIC.
55 0055 1 - Add the system node name to the header.
56 0056 1 - Add the number of days to each process CPU time.
57 0057 1 - Add buffered I/O to the direct I/O for each process.
```

Page 2
(1)

V03-005	LMP0140	L. Mark Pilant,	23-Aug-1983	23:29
		Add support for alphanumeric UICs.		
V03-004	GAS0117	Gerry Smith	12-Apr-1983	
		Instead of displaying MWAIT, display which resource the process is awaiting, or MUTEX if waiting for that.		
V03-003	CWH1002	CW Hobbs	25-Feb-1983	
		Use extended pids for the process ids.		
V03-002	GAS0107	Gerry Smith	8-Feb-1983	
		Collect all the qualifiers before making checks on whether or not any were set.		
V03-001	GAS00103		17-Jan-1983	
		Initialize the PCB before going into the PIX loop.		


```
78 0077 1
79 0078 1
80 0079 1  Include files
81 0080 1
82 0081 1
83 0082 1  LIBRARY 'SYSS$LIBRARY:LIB';          ! VAX/VMS system definitions
84 0083 1  REQUIRE 'SRC$:SHOWDEF';            ! SHOW common definitions
85 0182 1
86 0183 1  ! Define macro to make a string descriptor: sd_A
87 0184 1
88 0185 1  MACRO
89 M 0186 1      SD[A] =
90 0187 1          BIND %NAME('SD_',A) = $DESCRIPTOR(A)%;
91 0188 1
92 0189 1
93 0190 1  ! Define the flags for SHOW SYSTEM
94 0191 1
95 0192 1  MACRO
96 0193 1      sys$sv_proc = 0, 0, 1, 0%,
97 0194 1      sys$sv_subp = 0, 1, 1, 0%,
98 0195 1      sys$sv_net  = 0, 2, 1, 0%,
99 0196 1      sys$sv_batch = 0, 3, 1, 0%,
100 0197 1      sys$sv_full  = 0, 4, 1, 0%;
101 0198 1
102 0199 1
103 0200 1  ! Macros to define the layout of the data block to used to
104 0201 1  ! hold information on a particular process
105 0202 1
106 0203 1  MACRO
107 0204 1      d$l_pid      = 0, 0, 32, 0%,          ! Process ID
108 0205 1      d$l_owner    = 4, 0, 32, 0%,          ! Temp owner storage/process name length
109 0206 1      d$a_name     = 8, 0, 32, 0%,          ! Pointer to process name string
110 0207 1      d$l_state   = 12, 0, 32, 0%,         ! Process state
111 0208 1      d$l_pri     = 16, 0, 32, 0%,         ! Current priority
112 0209 1      d$l_iocnt  = 20, 0, 32, 0%,         ! Direct plus Buffered I/O count
113 0210 1      d$l_cputim  = 24, 0, 32, 0%,         ! CPU time
114 0211 1      d$l_pflts   = 28, 0, 32, 0%,         ! Page fault count
115 0212 1      d$l_pgcnt  = 32, 0, 32, 0%,         ! Global page count
116 0213 1      d$l_sts     = 36, 0, 32, 0%,         ! Status
117 0214 1      d$l_uic     = 40, 0, 32, 0%,         ! Process UIC
118 0215 1      d$l_lef     = 44, 0, 32, 0%,         ! Local event flag
119 0216 1      d$t_name    = 48, 0, 8, 0%,          ! Process name
120 0217 1
121 0218 1
122 0219 1  ! The following literal depends on D$t_NAME being the last field in the
123 0220 1  ! locked data area.
124 0221 1
125 0222 1  LITERAL d$k_length = $BYTEOFFSET(d$t_name) + pcb$s_lname;
126 0223 1
127 0224 1
128 0225 1  ! Define two bits in the data area, D$V_NETWORK and D$V_BATCH, which
129 0226 1  ! correspond to PCB$V_NETWORK and PCB$V_BATCH, except that they reference
130 0227 1  ! the process status from the PCB status longword, rather than from the
131 0228 1  ! beginning of the PCB.
132 0229 1
133 0230 1  MACRO
134 0231 1      d$sv_netwrk = 0, $BITPOSITION(pcb$sv_netwrk), $FIELDWIDTH(pcb$sv_netwrk), $EXTENSION(pcb$sv_netwrk)%;
```

SHOWSYSTEM
V04-000

J 14
16-Sep-1984 01:22:08
14-Sep-1984 12:09:48

VAX-11 Bliss-32 V4.0-742
[CLIUTL.SRC]SHOWSYS.B32;1

Page 4
(2)

; 135 0232 1 d\$y_batch = 0, \$BITPOSITION(pcb\$y_batch), \$FIELDWIDTH(pcb\$y_batch), \$EXTENSION(pcb\$y_batch)%;


```
137 0233 1 |
138 0234 1 | Construct a table of process states. THIS IS AN ORDERED TABLE.
139 0235 1 |
140 0236 1 LITERAL rsn_cnt = 14;
141 0237 1 OWN
142 0238 1     state_table : VECTOR[14]
143 0239 1     INITIAL (cstring('COLPG'),
144 0240 1             cstring('MUTEX'),
145 0241 1             cstring('CEF'),
146 0242 1             cstring('PFW'),
147 0243 1             cstring('LEF'),
148 0244 1             cstring('LEFO'),
149 0245 1             cstring('HIB'),
150 0246 1             cstring('HIBO'),
151 0247 1             cstring('SUSP'),
152 0248 1             cstring('SUSPO'),
153 0249 1             cstring('FPG'),
154 0250 1             cstring('COM'),
155 0251 1             cstring('COMO'),
156 0252 1             cstring('CUR')),
157 0253 1     rsn_table : VECTOR[rsn_cnt]
158 0254 1     INITIAL (cstring('RWAST'),
159 0255 1             cstring('RWMBX'),
160 0256 1             cstring('RWNPB'),
161 0257 1             cstring('RWPFF'),
162 0258 1             cstring('RWPAG'),
163 0259 1             cstring('RWBRK'),
164 0260 1             cstring('RWIMG'),
165 0261 1             cstring('RWQUO'),
166 0262 1             cstring('RWLCK'),
167 0263 1             cstring('RWSWP'),
168 0264 1             cstring('RWMPE'),
169 0265 1             cstring('RWMPB'),
170 0266 1             cstring('RWSGS'),
171 0267 1             cstring('RWCLU'));
172 0268 1
173 0269 1 |
174 0270 1 | This ASSUME macro makes sure that we have all the miscellaneous resource
175 0271 1 | waits in this table.
176 0272 1 |
177 0273 1 $ASSUME(rsn_cnt, EQL, rsn$_max-1)
178 0274 1 |
```

AST wait
Mailbox full
Non-paged pool
Page file full
Paged pool
Waiting for BROADCAST to finish
Image activation lock
Pooled quota
Lock ID data base
Swap file space
Modified page list empty
Modified page writer busy
SGS wait
Cluster translation wait

SHOWSYSTEM
V04-000

L 14
16-Sep-1984 01:22:08
14-Sep-1984 12:09:48

VAX-11 Bliss-32 V4.0-742
[CLIUTL.SRC]SHOWSYS.B32;1

Page 6
(4)

```
: 180      0275 1  |
: 181      0276 1  | Table of contents
: 182      0277 1  |
: 183      0278 1  |
: 184      0279 1 FORWARD ROUTINE
: 185      0280 1     show$system : NOVALUE,
: 186      0281 1     get_data,
: 187      0282 1     print_data : NOVALUE;
: 188      0283 1
: 189      0284 1 FORWARD
: 190      0285 1     lock_start : VECTOR[0],
: 191      0286 1     lock_end : VECTOR[0];
: 192      0287 1
: 193      0288 1 EXTERNAL
: 194      0289 1     scs$ga_localsb,
: 195      0290 1     sys$gq_version,
: 196      0291 1     exe$gl_abstim,
: 197      0292 1     sch$gl_pcbvec : REF VECTOR,
: 198      0293 1     sch$gl_maxpix;
: 199      0294 1
: 200      0295 1 EXTERNAL ROUTINE
: 201      0296 1     lib$get_vm,
: 202      0297 1     cli$present,
: 203      0298 1     show$write_line : NOVALUE;
```

```
! Local system block
! System version
! Time system has been up
! PCB vector
! Maximum process index count
```



```
0299 1 GLOBAL ROUTINE show$system : NOVALUE =
0300 2 BEGIN
0301 2
0302 2 ---
0303 2
0304 2 This is the driver routine for the SHOW SYSTEM function. The command
0305 2 qualifiers are gathered, scratch space is allocated, the data-gathering
0306 2 routine is called via $CMKRNL, and then the data is printed.
0307 2
0308 2 ---
0309 2
0310 2 LOCAL
0311 2     status,           ! General status return
0312 2     size,             ! Size of scratch area
0313 2     flags : $BBLOCK[2], ! Flags byte
0314 2     desc : VECTOR[2],  ! Argument list for calls
0315 2     data : VECTOR[2];  ! Address limits of scratch area
0316 2
0317 2
0318 2 Collect qualifiers.
0319 2
0320 2 flags[sys$V_full] = cli$present(%ASCID 'FULL');
0321 2 flags[sys$V_subp] = cli$present(%ASCID 'SUBPROCESS');
0322 2 flags[sys$V_net]  = cli$present(%ASCID 'NETWORK');
0323 2 flags[sys$V_batch] = cli$present(%ASCID 'BATCH');
0324 2 IF NOT (.flags[sys$V_subp] OR
0325 2         .flags[sys$V_net] OR
0326 2         .flags[sys$V_batch])
0327 2 THEN flags[sys$V_proc] = true;
0328 2
0329 2
0330 2 Allocate a scratch area in which to put data about the processes.
0331 2 The size of the scratch area is determined by taking the amount of
0332 2 bytes of information per process (D$K_LENGTH), multiplying that by
0333 2 the maximum number of processes in the system, and then adding a few
0334 2 pages for slop. The beginning and ending addresses of the area will
0335 2 be returned in DATA.
0336 2
0337 2 size = (.sch$gl_maxpix * d$k_length) + (3 * 512);
0338 2
0339 2 IF NOT (status = LIB$GET_VM(size,
0340 2                             data))
0341 2 THEN SIGNAL_STOP(show$_insvirmem, 0, .status);
0342 2 data[1] = .data[0] + .size - 1;
0343 2
0344 2
0345 2 Lock the first page of the scratch area, and the code that runs at elevated
0346 2 IPL, into the process working set.
0347 2
0348 2 desc[0] = lock_start;
0349 2 desc[1] = lock_end;
0350 2 IF NOT (status = $LKWSET(INADR = desc))
0351 2 THEN SIGNAL_STOP(.status);
0352 2
0353 2 desc[0] = .data[0];
0354 2 desc[1] = .data[0] + d$k_length;
0355 2 IF NOT (status = $LKWSET(INADR = desc))
```



```
262 0356 2 THEN SIGNAL_STOP(.status);
263 0357
264 0358
265 0359
266 0360
267 0361
268 0362
269 0363
270 0364
271 P 0365 IF NOT (status = $CMKRN (ROUTIN = get_data,
272 0366 ARGST = desc))
273 0367 THEN
274 0368 BEGIN
275 0369 SIGNAL(.status);
276 0370 RETURN;
277 0371 END;
278 0372
279 0373
280 0374
281 0375
282 0376
283 0377
284 0378
285 0379 1 RETURN;
END;
```

Call the data-gathering routine in kernel mode, passing the address limits as an argument.

desc[0] = 2;
desc[1] = data;
desc[2] = flags;

Format and print the data.

print_data(data, flags);

! End of show\$error

.TITLE SHOWSYSTEM
.IDENT \V04-000\
.PSECT \$SPLITS,NOWRT,NOEXE,2

47	50	4C	4F	43	05	00000	P.AAA:	.ASCII	<5>\COLPG\
58	45	54	55	4D	05	00006	P.AAB:	.ASCII	<5>\MUTEX\
		46	45	43	03	0000C	P.AAC:	.ASCII	<3>\CEF\
		57	46	50	03	00010	P.AAD:	.ASCII	<3>\PFW\
		46	45	4C	03	00014	P.AAE:	.ASCII	<3>\LEF\
	4F	46	45	4C	04	00018	P.AAF:	.ASCII	<4>\LEFO\
		42	49	48	03	0001D	P.AAG:	.ASCII	<3>\HIB\
	4F	42	49	48	04	00021	P.AAH:	.ASCII	<4>\HIBO\
	50	53	55	53	04	00026	P.AAI:	.ASCII	<4>\SUSP\
4F	50	53	55	53	05	0002B	P.AAJ:	.ASCII	<5>\SUSPO\
		47	50	46	03	00031	P.AAK:	.ASCII	<3>\FPG\
		4D	4F	43	03	00035	P.AAL:	.ASCII	<3>\COM\
	4F	4D	4F	43	04	00039	P.AAM:	.ASCII	<4>\COMO\
		52	55	43	03	0003E	P.AAN:	.ASCII	<3>\CUR\
54	53	41	57	52	05	00042	P.AAO:	.ASCII	<5>\RWAST\
58	42	4D	57	52	05	00048	P.AAP:	.ASCII	<5>\RWMBX\
47	50	4E	57	52	05	0004E	P.AAQ:	.ASCII	<5>\RWNPG\
46	46	50	57	52	05	00054	P.AAR:	.ASCII	<5>\RWPFF\
47	41	50	57	52	05	0005A	P.AAS:	.ASCII	<5>\RWPAG\
4B	52	42	57	52	05	00060	P.AAT:	.ASCII	<5>\RWBRK\
47	4D	49	57	52	05	00066	P.AAU:	.ASCII	<5>\RWIMG\
4F	55	51	57	52	05	0006C	P.AAV:	.ASCII	<5>\RWQUO\
4B	43	4C	57	52	05	00072	P.AAW:	.ASCII	<5>\RWLCK\
50	57	53	57	52	05	00078	P.AAX:	.ASCII	<5>\RWSWP\
45	50	4D	57	52	05	0007E	P.AAY:	.ASCII	<5>\RWMP\
42	50	4D	57	52	05	00084	P.AAZ:	.ASCII	<5>\RWMPB\


```
53 43 53 57 52 05 0008A P.ABA: .ASCII <5>\RWSCS\
55 4C 43 57 52 05 00090 P.ABB: .ASCII <5>\RWCLU\
00096 .BLKB 2
4C 4C 55 46 00098 P.ABD: .ASCII \FULL\
010E0004 0009C P.ABC: .LONG 17694724
00000000 000A0 .ADDRESS P.ABD
00 00 53 53 45 43 4F 52 50 42 55 53 000A4 P.ABF: .ASCII \SUBPROCESS\<0><0>
010E000A 000B0 P.ABE: .LONG 17694730
00000000 000B4 .ADDRESS P.ABF
00 4B 52 4F 57 54 45 4E 000B8 P.ABH: .ASCII \NETWORK\<0>
010E0007 000C0 P.ABG: .LONG 17694727
00000000 000C4 .ADDRESS P.ABH
00 00 00 48 43 54 41 42 000C8 P.ABJ: .ASCII \BATCH\<0><0><0>
010E0005 000D0 P.ABI: .LONG 17694725
00000000 000D4 .ADDRESS P.ABJ
```

.PSECT \$OWNS,NOEXE,2

```
00000000' 00000000' 00000000' 00000000' 00000000' 00000000' 00000 STATE_TABLE:
00000000' 00000000' 00000000' 00000000' 00000000' 00000000' 00018 .ADDRESS P.AAA, P.AAB, P.AAC, P.AAD, P.AAE, -
00000000' 00000000' 00000000' 00000000' 00000000' 00000000' 00030 P.AAF, P.AAG, P.AAH, P.AAI, P.AAJ, P.AAK, -
00000000' 00000000' 00000000' 00000000' 00000000' 00000000' 00038 RSN_TABLE: P.AAL, P.AAM, P.AAN
00000000' 00000000' 00000000' 00000000' 00000000' 00000000' 00050 .ADDRESS P.AAO, P.AAP, P.AAQ, P.AAR, P.AAS, -
00000000' 00000000' 00000000' 00000000' 00000000' 00000000' 00068 P.AAT, P.AAU, P.AAV, P.AAW, P.AAX, P.AAY, -
P.AAZ, P.ABA, P.ABB
```

```
.EXTRN SC$SGA_LOCALSB, SY$SGQ_VERSION
.EXTRN EX$SGL_ABSTIM, SCH$SGL_PCBVEC
.EXTRN SCH$SGL_MAXPIX, LIB$GET_VM
.EXTRN CL$SPRESENT, SHOW$WRITE_LINE
.EXTRN SY$SLKWSET, SY$SCMKRNL
```

.PSECT \$CODE\$,NOWRT,2

```
04 AE 01 00000000G 003C 00000 .ENTRY SHOW$SYSTEM, Save R2,R3,R4,R5 0299
55 00000000G 00 9E 00002 MOVAB SY$SLKWSET, R5
54 00000000G 00 9E 00009 MOVAB LIB$STOP, R4
53 00000000G 00 9E 00010 MOVAB CL$SPRESENT, R3
5E 18 C2 00017 SUBL2 #24, SP
0000' CF 9F 0001A PUSHAB P.ARC 0320
63 01 FB 0001E CALLS #1, CL$SPRESENT
04 50 FO 00021 INSV R0, #4, #1, FLAGS
0000' CF 9F 00027 PUSHAB P.ABE 0321
63 01 FB 0002B CALLS #1, CL$SPRESENT
04 50 FO 0002E INSV R0, #1, #1, FLAGS
0000' CF 9F 00034 PUSHAB P.ABG 0322
63 01 FB 00038 CALLS #1, CL$SPRESENT
04 50 FO 0003B INSV R0, #2, #1, FLAGS
0000' CF 9F 00041 PUSHAB P.ABI 0323
63 01 FB 00045 CALLS #1, CL$SPRESENT
04 50 FO 00048 INSV R0, #3, #1, FLAGS
0E 01 EO 0004E BBS #1, FLAGS, 1$ 0324
09 04 AE 02 EO 00053 BBS #2, FLAGS, 1$ 0325
04 04 AE 03 EO 00058 BBS #3, FLAGS, 1$ 0326
04 04 AE 01 88 0005D BISB2 #1, FLAGS 0327
50 00000000G 00 06 78 00061 1$: ASHL #6, SCH$GL_MAXPIX, R0 0337
```


		6E	0600	CO	9E	00069	MOVAB	1536(R0), SIZE		
			08	AE	9F	0006E	PUSHAB	DATA	...	0339
			04	AE	9F	00071	PUSHAB	SIZE		
	00000000G	00		02	FB	00074	CALLS	#2, LIB\$GET_VM		
		52		50	DO	0007B	MOVL	R0, STATUS		
		0D		52	E8	0007E	BLBS	STATUS, 2\$		
				52	DD	00081	PUSHL	STATUS	...	0341
				7E	D4	00083	CLRL	-(SP)		
			007812F2	8F	DD	00085	PUSHL	#7869170		
		64		03	FB	0008B	CALLS	#3, LIB\$STOP		
50	08	AE		6E	C1	0008E	ADDL3	SIZE, DATA, R0	...	0342
	0C	AE	FF	A0	9E	00093	MOVAB	-1(R0), DATA+4		
	10	AE	0000V	CF	9E	00098	MOVAB	LOCK_START, DESC	...	0348
	14	AE	0000V	CF	9E	0009E	MOVAB	LOCK_END, DESC+4	...	0349
				7E	7C	000A4	CLRQ	-(SP)	...	0350
			18	AE	9F	000A6	PUSHAB	DESC		
		65		03	FB	000A9	CALLS	#3, SYSS\$LKWSET		
		52		50	DO	000AC	MOVL	R0, STATUS		
		05		52	E8	000AF	BLBS	STATUS, 3\$		
				52	DD	000B2	PUSHL	STATUS	...	0351
		64		01	FB	000B4	CALLS	#1, LIB\$STOP		
14	AE	10	08	AE	DO	000B7	MOVL	DATA, DESC	...	0353
		08	AE	00000040	8F	C1	ADDL3	#64, DATA, DESC+4	...	0354
				7E	7C	000C6	CLRQ	-(SP)	...	0355
			18	AE	9F	000C8	PUSHAB	DESC		
		65		03	FB	000CB	CALLS	#3, SYSS\$LKWSET		
		52		50	DO	000CE	MOVL	R0, STATUS		
		05		52	E8	000D1	BLBS	STATUS, 4\$		
				52	DD	000D4	PUSHL	STATUS	...	0356
		64		01	FB	000D6	CALLS	#1, LIB\$STOP		
		10	08	02	DO	000D9	MOVL	#2, DESC	...	0362
		14	AE	AE	9E	000DD	MOVAB	DATA, DESC+4	...	0363
			04	AE	9E	000E2	MOVAB	FLAGS, DESC+8	...	0364
			10	AE	9F	000E6	PUSHAB	DESC	...	0366
			0000V	CF	9F	000E9	PUSHAB	GET_DATA		
	00000000G	00		02	FB	000ED	CALLS	#2, SYSS\$CMKRNL		
		52		50	DO	000F4	MOVL	R0, STATUS		
		0A		52	E8	000F7	BLBS	STATUS, 5\$		
				52	DD	000FA	PUSHL	STATUS	...	0369
	00000000G	00		01	FB	000FC	CALLS	#1, LIB\$SIGNAL		
				04	AE	00103	RET		...	0368
			04	AE	9F	00104	PUSHAB	FLAGS	...	0376
			0C	AE	9F	00107	PUSHAB	DATA		
	0000V	CF		02	FB	0010A	CALLS	#2, PRINT_DATA		
				04	AE	0010F	RET		...	0379

; Routine Size: 272 bytes, Routine Base: \$CODE\$ + 0000


```
287 0380 1 OWN lock_start : VECTOR[0] PSECT ($CODE$);      ! Beginning of locked code
288 0381 1 ROUTINE get_data (data, flags) =
289 0382 2 BEGIN
290 0383 2
291 0384 2 ---
292 0385 2
293 0386 2 This routine executes in KERNEL mode. It scans all the processes in the
294 0387 2 system, gathering information on them.
295 0388 2
296 0389 2 Inputs
297 0390 2     DATA -- address of the scratch area
298 0391 2     FLAGS -- options longword, to tell what kind of processes are desired
299 0392 2
300 0393 2 Outputs
301 0394 2     DATA -- will contain information on the processes
302 0395 2
303 0396 2 ---
304 0397 2
305 0398 2 MAP
306 0399 2     data : REF VECTOR,
307 0400 2     flags : REF $BBLOCK;
308 0401 2
309 0402 2 REGISTER
310 0403 2     locked : REF $BBLOCK,      ! Pointer to locked page
311 0404 2     scratch : REF $BBLOCK,     ! Pointer to scratch area
312 0405 2     pcb : REF $BBLOCK,        ! Pointer to PCB
313 0406 2     null,                     ! Null process PCB address
314 0407 2     pix;                       ! Process index
315 0408 2
316 0409 2
317 0410 2 The first page of the scratch area is locked, so that it can be accessed at
318 0411 2 elevated IPL. This locked portion will be a temporary storage place for
319 0412 2 information about one process at a time. The remainder of the scratch area
320 0413 2 will contain information on the processes which are to be displayed.
321 0414 2 Set up these areas so that they can be addressed easily.
322 0415 2
323 0416 2 locked = .data[0];             ! Point to locked area
324 0417 2 scratch = .data[0] + d$length; ! Scratch area is just beyond
325 0418 2                               ! the locked data
326 0419 2
327 0420 2 null = pcb = .sch$gl_pcbvec[0]; ! Save address of NULL PCB
328 0421 2
329 0422 2 INCR pix FROM 0 TO .sch$gl_maxpix
330 0423 2 DO
331 0424 2     BEGIN
332 0425 2     SET_IPL(IPL$ SYNCH);      ! Raise IPL
333 0426 2     IF .pix EQL 0
334 0427 2     OR (pcb = .sch$gl_pcbvec[.pix]) NEQ .null
335 0428 2     THEN
336 0429 2         BEGIN
337 0430 2         locked[d$l_pid] = .pcb[pcb$l_epid]; ! Use the extended pid
338 0431 2         locked[d$l_owner] = .pcb[pcb$l_owner];
339 0432 2         locked[d$l_uic] = .pcb[pcb$l_uic];
340 0433 2         locked[d$l_state] = .pcb[pcb$w_state];
341 0434 2         locked[d$l_pri] = .pcb[pcb$b_pri];
342 0435 2         locked[d$l_pgcnt] = .pcb[pcb$w_ppgcnt] + .pcb[pcb$w_gpgcnt];
343 0436 2         locked[d$l_lef] = .pcb[pcb$l_efwm];
```



```
! End of GET_DATA
```

OFFC 00000 GET_DATA:

PC	OP	OP2	OP3	OP4	OP5	OP6	OP7	OP8	OP9	OP10	OP11	OP12	OP13	OP14	OP15	OP16	OP17	OP18	OP19	OP20	OP21	OP22	OP23	OP24	OP25	OP26	OP27	OP28	OP29	OP30	OP31	OP32	OP33	OP34	OP35	OP36	OP37	OP38	OP39	OP40	OP41	OP42	OP43	OP44	OP45	OP46	OP47	OP48	OP49	OP50	OP51	OP52	OP53	OP54	OP55	OP56	OP57	OP58	OP59	OP60	OP61	OP62	OP63	OP64	OP65	OP66	OP67	OP68	OP69	OP70	OP71	OP72	OP73	OP74	OP75	OP76	OP77	OP78	OP79	OP80	OP81	OP82	OP83	OP84	OP85	OP86	OP87	OP88	OP89	OP90	OP91	OP92	OP93	OP94	OP95	OP96	OP97	OP98	OP99	OP100	OP101	OP102	OP103	OP104	OP105	OP106	OP107	OP108	OP109	OP110	OP111	OP112	OP113	OP114	OP115	OP116	OP117	OP118	OP119	OP120	OP121	OP122	OP123	OP124	OP125	OP126	OP127	OP128	OP129	OP130	OP131	OP132	OP133	OP134	OP135	OP136	OP137	OP138	OP139	OP140	OP141	OP142	OP143	OP144	OP145	OP146	OP147	OP148	OP149	OP150	OP151	OP152	OP153	OP154	OP155	OP156	OP157	OP158	OP159	OP160	OP161	OP162	OP163	OP164	OP165	OP166	OP167	OP168	OP169	OP170	OP171	OP172	OP173	OP174	OP175	OP176	OP177	OP178	OP179	OP180	OP181	OP182	OP183	OP184	OP185	OP186	OP187	OP188	OP189	OP190	OP191	OP192	OP193	OP194	OP195	OP196	OP197	OP198	OP199	OP200	OP201	OP202	OP203	OP204	OP205	OP206	OP207	OP208	OP209	OP210	OP211	OP212	OP213	OP214	OP215	OP216	OP217	OP218	OP219	OP220	OP221	OP222	OP223	OP224	OP225	OP226	OP227	OP228	OP229	OP230	OP231	OP232	OP233	OP234	OP235	OP236	OP237	OP238	OP239	OP240	OP241	OP242	OP243	OP244	OP245	OP246	OP247	OP248	OP249	OP250	OP251	OP252	OP253	OP254	OP255	OP256	OP257	OP258	OP259	OP260	OP261	OP262	OP263	OP264	OP265	OP266	OP267	OP268	OP269	OP270	OP271	OP272	OP273	OP274	OP275	OP276	OP277	OP278	OP279	OP280	OP281	OP282	OP283	OP284	OP285	OP286	OP287	OP288	OP289	OP290	OP291	OP292	OP293	OP294	OP295	OP296	OP297	OP298	OP299	OP300	OP301	OP302	OP303	OP304	OP305	OP306	OP307	OP308	OP309	OP310	OP311	OP312	OP313	OP314	OP315	OP316	OP317	OP318	OP319	OP320	OP321	OP322	OP323	OP324	OP325	OP326	OP327	OP328	OP329	OP330	OP331	OP332	OP333	OP334	OP335	OP336	OP337	OP338	OP339	OP340	OP341	OP342	OP343	OP344	OP345	OP346	OP347	OP348	OP349	OP350	OP351	OP352	OP353	OP354	OP355	OP356	OP357	OP358	OP359	OP360	OP361	OP362	OP363	OP364	OP365	OP366	OP367	OP368	OP369	OP370	OP371	OP372	OP373	OP374	OP375	OP376	OP377	OP378	OP379	OP380	OP381	OP382	OP383	OP384	OP385	OP386	OP387	OP388	OP389	OP390	OP391	OP392	OP393	OP394	OP395	OP396	OP397	OP398	OP399	OP400	OP401	OP402	OP403	OP404	OP405	OP406	OP407	OP408	OP409	OP410	OP411	OP412	OP413	OP414	OP415	OP416	OP417	OP418	OP419
----	----	-----	-----	-----	-----	-----	-----	-----	-----	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------

SHOWSYSTEM
V04-000

F 15
16-Sep-1984 01:22:08 VAX-11 Bliss-32 V4.0-742
14-Sep-1984 12:09:48 [CLIUTL.SRC]SHOWSYS.B32;1

Page 13
(6)

			50	24	A8	D0	0005D	MOVL	36(PCB), R0	:	0437
		24	A6		50	D0	00061	MOVL	R0, 36(LOCKED)	:	
			15		50	E9	00065	BLBC	R0, 3\$:	
			50	6C	A8	D0	00068	MOVL	108(PCB), R0	:	0440
14	A6	54	A0	58	A0	C1	0006C	ADDL3	88(R0), 84(R0), 20(LOCKED)	:	0441
		1C	A6	4C	A0	D0	00073	MOVL	76(R0), 28(LOCKED)	:	0442
		18	A6	38	A0	D0	00078	MOVL	56(R0), 24(LOCKED)	:	0443
30	A6	70	A8		10	28	0007D	3\$: MOVC3	#16, 112(PCB), 48(LOCKED)	:	0447
			12		00	DA	00083	MTPR	#0, #18	:	0448
			1E	08	BC	E8	00086	BLBS	@FLAGS, 6\$:	0450
	05	08	BC		01	E1	0008A	BBC	#1, @FLAGS, 4\$:	0451
				04	A6	D5	0008F	TSTL	4(LOCKED)	:	
					14	12	00092	BNEQ	6\$:	
	05	08	BC		03	E1	00094	4\$: BBC	#3, @FLAGS, 5\$:	0452
	0A	25	A6		06	E0	00099	BBS	#6, 37(LOCKED), 6\$:	
	0E	08	BC		02	E1	0009E	5\$: BBC	#2, @FLAGS, 7\$:	0453
	09	26	A6		05	E1	000A3	BBC	#5, 38(LOCKED), 7\$:	
	67		66	0040	8F	28	000A8	6\$: MOVC3	#64, (LOCKED), (SCRATCH)	:	0454
			57		53	D0	000AE	MOVL	R3, SCRATCH	:	
			12		00	DA	000B1	7\$: MTPR	#0, #18	:	0457
FF64			01	00000000G	00	F1	000B4	8\$: ACBL	SCH\$GL_MAXPIX, #1, PIX, 1\$:	0422
	5A		50		01	D0	000BE	MOVL	#1, R0	:	0460
					04	000C1		RET		:	0461

; Routine Size: 194 bytes, Routine Base: \$CODE\$ + 0110

; 369 0462 1 OWN lock_end : VECTOR[0] PSECT (\$CODE\$); ! End of locked code


```
371 0463 1 ROUTINE print_data (data,flags) : NOVALUE =
372 0464 2 BEGIN
373 0465 3
374 0466 4 ---
375 0467 5
376 0468 6 This routine prints the data contained in DATA, the scratch area
377 0469 7
378 0470 8 Inputs
379 0471 9 DATA -- scratch area, contains the process information (didn't I
380 0472 10 just say that?)
381 0473 11
382 0474 12 FLAGS -- contains the bits set for the qualifiers. It is specifically
383 0475 13 used in this routine to determine if the /FULL qualifier is
384 0476 14 set.
385 0477 15
386 0478 16 Outputs
387 0479 17 The process data is printed.
388 0480 18
389 0481 19 --
390 0482 20
391 0483 21 MAP
392 0484 22 data : REF VECTOR,
393 0485 23 flags: REF $BBLOCK;
394 0486 24
395 0487 25 LOCAL
396 0488 26 scratch : REF $BBLOCK,
397 0489 27 status,
398 0490 28 time : VECTOR[2],
399 0491 29 desc : VECTOR[2],
400 0492 30 proctim : VECTOR[4],
401 0493 31 arglist : VECTOR[4],
402 0494 32 sysnodebuf : VECTOR[16,BYTE],
403 0495 33 desc_sysnode: $BBLOCK[DSC$_S_BLN],
404 0496 34
405 0497 35 trnlmlst : $ITMLST_DECL (ITEMS = 1);
406 0498 36
407 0499 37
408 0500 38 ! Set up string descriptor to find the logical name system table
409 0501 39
410 0502 40 sd ('LNMS$SYSTEM');
411 0503 41
412 0504 42
413 0505 43 ! Set up the scratch area, which contains all the data about the processes.
414 0506 44 The data is located beyond the locked segment.
415 0507 45
416 0508 46 scratch = .data[0] + d$k_length;
417 0509 47
418 0510 48
419 0511 49 ! If there is no data in the scratch area, then simply return.
420 0512 50
421 0513 51 IF .scratch[d$l_pid] EQL 0
422 0514 52 THEN RETURN;
423 0515 53
424 0516 54
425 0517 55 ! Determine the time the system has been up. This is done using the
426 0518 56 value of EXE$GL_ABSTIM, multiplying it by the right constant and
427 0519 57 handing it to $ASCTIM.
```



```

428      0520      2 !
429      0521      2 EMUL(%REF(.exe$gl_abstim), %REF(-10000000), %REF(0), time); ! Get the uptime
430      0522      2 desc[0] = d$length; ! Set up a descriptor pointing
431      0523      2 desc[1] = .data[0]; ! to the locked (scratch) area
432      0524      2 IF NOT (status = $ASCTIM(TIMADR = time, ! Convert the uptime
433      0525      2 ! TIMBUF = desc, ! to ASCII, storing here,
434      0526      2 ! TIMLEN = desc, ! put length here,
435      0527      2 ! CVTFLG = 0)) ! and give full date and time
436      0528      2 THEN
437      0529      2 BEGIN
438      0530      2 SIGNAL(.status);
439      0531      2 RETURN;
440      0532      2 END;
441      0533      2
442      0534      2 ! Initialize the descriptor for system node.
443      0535      2
444      0536      2 desc_sysnode = 16; ! Address of space containing
445      0537      2 desc_sysnode[dsc$a_pointer] = sysnodebuf; ! system node
446      0538      2
447      0539      2 ! Initialize item list used to find system node
448      0540      2
449      0541      2 $ITMLST_INIT (ITMLST = trnlmlst, ! Pre-declared address for item list
450      0542      2 (ITMCO = LNM$STRING, ! Logical name translation string
451      0543      2 ! is to be obtained
452      0544      2 ! BUFADR = sysnodebuf, ! Address to put system node
453      0545      2 ! BUFSIZ = 16, ! Length of BUFADR
454      0546      2 ! RETLEN = desc_sysnode) ! Length of returned node
455      0547      2 );
456      0548      2
457      0549      2 ! Get system node. Documented in Specification for VMS Logical Name Extension
458      0550      2
459      0551      2 IF NOT (status = $STRNLNM
460      0552      2 (ATTR = %REF(lnm$m_case_blind), ! Letter case makes no difference
461      0553      2 TABNAM = SD_LNM$SYSTEM, ! Logical name table to be searched
462      0554      2 LOGNAM = $DESCRIPTOR('SYS$NODE'), ! What to translate
463      0555      2 ACMODE = %REF(PSL$C_EXEC), ! Access mode to use
464      0556      2 ITMLST = trnlmlst ! Predefined item list
465      0557      2 )
466      0558      2 )
467      0559      2 THEN desc_sysnode[dsc$w_length] = 0
468      0560      2 ELSE
469      0561      2
470      0562      2 ! Strip leading underscore and trailing colons, if either, from node name
471      0563      2
472      0564      2 BEGIN
473      0565      2 IF .sysnodebuf[0] EQL '_'
474      0566      2 THEN
475      0567      2 BEGIN
476      0568      2 desc_sysnode[dsc$w_length] = .desc_sysnode[dsc$w_length] - 1;
477      0569      2 desc_sysnode[dsc$a_pointer] = .desc_sysnode[dsc$a_pointer] + 1;
478      0570      2 END;
479      0571      2
480      0572      2 INCRU I FROM 0 TO 1
481      0573      2 DO IF NOT CH$FAIL(CH$FIND_CH(.desc_sysnode[dsc$w_length],
482      0574      2 .desc_sysnode[dsc$a_pointer], ':'))
483      0575      2 THEN desc_sysnode[dsc$w_length] = .desc_sysnode[dsc$w_length] - 1;
484      0576      2 END;
```



```
485 0577 2 |
486 0578 2 |
487 0579 2 | Set up the $FAOL parameter list, with the addresses of the descriptors of
488 0580 2 | the system version, the system node name, and the uptime.
489 0581 2 |
490 0582 2 arglist[0] = UPLIT(4, sys$gq_version); | Version number is 4 bytes
491 0583 2 arglist[1] = desc_sysnode; | System node name string desc
492 0584 2 arglist[2] = 0; | Zero to get current date
493 0585 2 arglist[3] = desc; | Uptime string desc
494 0586 2 desc[0] = .desc[0] - 3; | Get rid of trailing ".00"
495 0587 2 |
496 0588 2 |
497 0589 2 | Now format and print the header lines.
498 0590 2 |
499 0591 2 show$write_line(%ASCII 'VAX/VMS !AS on node !AS !%D Uptime !AS',
500 0592 2 arglist,
501 0593 2 %ASCII ' Pid Process Name State Pri I/O CPU Page flts Ph.Mem',
502 0594 2 0);
503 0595 2 |
504 0596 2 |
505 0597 2 | Loop thru the scratch area, formatting and outputting the data one process
506 0598 2 | at a time. The data block for each process is set up in as an ordered
507 0599 2 | sequence of longwords, in the order of the arguments to $FAOL, so that the
508 0600 2 | data block itself can be used as the parameter list to $FAOL. All that is
509 0601 2 | required is some minor fixup.
510 0602 2 |
511 0603 2 WHILE .scratch[d$l_pid] NEQ 0 | Loop thru all processes
512 0604 2 DO
513 0605 2 BEGIN
514 0606 2 |
515 0607 2 | Get the state.
516 0608 2 |
517 0609 2 IF .scratch[d$l_state] GEQ sch$c_colpg
518 0610 2 AND .scratch[d$t_state] LEQ sch$c_cur
519 0611 2 THEN
520 0612 4 BEGIN
521 0613 4 IF .scratch[d$l_state] EQL sch$c_mwait
522 0614 4 AND .scratch[d$t_lef] GEQ 0
523 0615 4 THEN
524 0616 5 BEGIN
525 0617 5 IF .scratch[d$l_lef] GEQ 1
526 0618 5 AND .scratch[d$t_lef] LEQ rsn_cnt
527 0619 5 THEN scratch[d$l_state] = .rsn_table[.scratch[d$l_lef] - 1]
528 0620 5 ELSE scratch[d$l_state] = cstring('RWUNK');
529 0621 5 END
530 0622 4 ELSE scratch[d$l_state] = .state_table[.scratch[d$l_state] - 1];
531 0623 4 END
532 0624 2 ELSE scratch[d$l_state] = cstring('UNK');
533 0625 2 |
534 0626 2 |
535 0627 2 | If the owner field is not empty, then the process is a subprocess;
536 0628 2 | otherwise, check the status bits for a network or batch process.
537 0629 2 |
538 0630 2 status = .scratch[d$l_sts]; | Save the status
539 0631 2 IF .scratch[d$l_owner] NEQ 0
540 0632 2 THEN scratch[d$t_sts] = cstring('S')
541 0633 2 ELSE IF .$BLOCK[scratch[d$l_sts], d$v_netwrk]
```



```
542 0634 THEN scratch[d$l_sts] = cstring('N')
543 0635 ELSE IF $.BBLCK[scratch[d$l_sts], d$V_batch]
544 0636 THEN scratch[d$l_sts] = cstring('B')
545 0637 ELSE scratch[d$l_sts] = cstring(' ');
546 0638
547 0639
548 0640 Convert the priority from the internal format to the external format.
549 0641
550 0642 scratch[d$l_pri] = 31 - .scratch[d$l_pri];
551 0643
552 0644
553 0645 The process name string should be converted from the counted string
554 0646 format used in the PCB to a length/address descriptor, so that $FAO
555 0647 can print non-alphanumerics as periods. To do this, use the OWNER
556 0648 field in the scratch area as the count, and point to the beginning
557 0649 of the saved string, instead of the count.
558 0650
559 0651 scratch[d$l_owner] = .scratch[d$t_name];
560 0652 scratch[d$a_name] = scratch[d$t_name] + 1;
561 0653
562 0654
563 0655 Multiply the CPU time for the process by -100000, setting it up for
564 0656 conversion into ASCII form.
565 0657
566 0658 IF $.status
567 0659 THEN
568 0660 BEGIN
569 0661 EMUL(scratch[d$l_cputim], %REF(-100000), %REF(0), time);
570 0662 desc[0] = 16;
571 0663 desc[1] = proctim;
572 0664 ! Desc will contain the address
573 0665 IF NOT (status = $ASCTIM(TIMADR = time, ! of where the actual time is.
574 0666 TIMBUF = desc, ! Convert the uptime
575 0667 TIMLEN = desc, ! to ASCII, storing here,
576 0668 CVTFLG = 0)) ! put length here,
577 0669 ! and give full date and time
578 0670 THEN
579 0671 BEGIN
580 0672 SIGNAL($.status);
581 0673 RETURN;
582 0674 ELSE
583 0675 scratch[d$l_cputim] = desc;
584 0676 END;
585 0677
586 0678 Now produce the line of text and output it.
587 0679
588 0680 IF $.status
589 0681 THEN
590 0682 BEGIN
591 0683 IF $.flags[sys$V_full]
592 0684 THEN show$write_line(%ASCID '!8XL !15AF !5AC !3UB!9UL!AS !9UL !5UW !AC!/ !%I',
593 0685 .scratch)
594 0686 ELSE show$write_line(%ASCID '!8XL !15AF !5AC !3UB!9UL!AS !9UL !5UW !AC',
595 0687 .scratch)
596 0688 END
597 0689 ELSE
598 0690 BEGIN
```



```
599 0691 4 IF .flags[sys$u_full]
600 0692 4 THEN show$write_line(%ASCII '!8XL !15AF !5AC !3UB!3(+ -- swapped out -- !5UW !
601 0693 4 .scratch)
602 0694 4 ELSE show$write_line(%ASCII '!8XL !15AF !5AC !3UB!3(+ -- swapped out -- !5UW !
603 0695 4 .scratch)
604 0696 4 END;
605 0697 4
606 0698 4 Adjust the scratch pointer to point to the next block of process info.
607 0699 4
608 0700 4 scratch = .scratch + d$k_length;
609 0701 4 END;
610 0702 4
611 0703 4 RETURN;
612 0704 1 END;
```

! End of PRINT_DATA

```
001D2 .BLKB 2
001D4 LOCK_END: .BLKB 0
.PSECT $PLITS$,NOWRT,NOEXE,2
4D 45 54 53 59 53 24 4D 4E 4C 000D8 P.ABL: .ASCII \LNM$SYSTEM\
000E2 .BLKB 2
0000000A 000E4 P.ABK: .LONG 10
00000000' 000E8 .ADDRESS P.ABL
45 44 4F 4E 24 53 59 53 000EC P.ABN: .ASCII \SYSS$NODE\
00000008 000F4 P.ABM: .LONG 8
00000000' 000F8 .ADDRESS P.ABN
00000004 000FC P.ABO: .LONG 4
00000000G 00100 .ADDRESS SYSS$Q VERSION
6E 6F 20 20 53 41 21 20 53 4D 56 2F 58 41 56 00104 P.ABQ: .ASCII \VAX/VMS !AS on node !AS !%D Uptime !A\
20 20 44 25 21 20 53 41 21 20 65 64 6F 6E 20 00113
41 21 20 65 6D 69 74 70 55 20 00122
00 00 00 53 0012C
010E0029 00130 P.ABP: .ASCII \S\<0><0><0>
00000000' 00134 .LONG 17694761
73 65 63 6F 72 50 20 20 20 20 64 69 50 20 20 00138 P.ABS: .ADDRESS P.ABQ
65 74 61 74 53 20 20 20 20 65 6D 61 4E 20 73 00147 P.ABS: .ASCII \ Pid Process Name State Pri \
20 55 50 43 20 20 20 20 20 20 69 72 50 20 20 00156
73 74 6C 66 20 65 67 61 50 20 20 20 20 20 20 00160 .ASCII \ I/O CPU Page flts Ph.Mem\<0>
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 0016F
00 6D 65 4D 2E 68 50 20 0017E
00 00 00186
010E004D 00188 P.ABR: .ASCII <0><0>
00000000' 0018C .LONG 17694797
4B 4E 55 57 52 05 00190 P.ABT: .ADDRESS P.ABS
4B 4E 55 03 00196 P.ABU: .ASCII <5>\RWUNK\
53 01 0019A P.ABV: .ASCII <3>\UNK\
4E 01 0019C P.ABW: .ASCII <1>\S\
42 01 0019E P.ABX: .ASCII <1>\N\
20 01 001A0 P.ABY: .ASCII <1>\B\
001A2 P.ABY: .ASCII <1>\ \
43 41 35 21 20 46 41 35 31 21 20 4C 58 38 21 001A4 P.ACA: .BLKB 2
21 20 53 41 21 4C 55 39 21 42 55 33 21 20 20 001A6 P.ACA: .ASCII \!8XL !15AF !5AC !3UB!9UL!AS !9UL !5UW \
20 57 55 35 21 20 20 4C 55 39 001B3
001C2
```


SHOWSYSTEM
V04-000

L 15
16-Sep-1984 01:22:08 VAX-11 Bliss-32 V4.0-742
14-Sep-1984 12:09:48 [CLIUTL.SRC]SHOWSYS.B32;1

Page 19
(7)

```
20 20 20 20 20 20 20 20 20 2F 21 43 41 21 20 001CC .ASCII \ !AC!/ !XI\<0><0>
00 00 49 25 21 001DB
010E003A 001E0 P.ABZ: .LONG 17694778
00000000 001E4 .ADDRESS P.ACA
43 41 35 21 20 46 41 35 31 21 20 4C 58 38 21 001E8 P.ACC: .ASCII \!8XL !15AF !5AC !3UB!9UL!AS !9UL !5UW \
21 20 53 41 21 4C 55 39 21 42 55 33 21 20 20 001F7
20 57 55 35 21 20 20 4C 55 39 00206
43 41 35 21 20 46 41 35 31 21 20 4C 58 38 21 00210
20 20 20 20 29 28 28 33 21 42 55 33 21 20 20 00214 P.ACB: .ASCII \ !AC\
61 77 73 20 20 20 20 20 20 20 00218 P.ACB: .LONG 17694764
20 20 20 20 20 20 74 75 6F 20 20 64 65 70 70 0021C P.ACE: .ADDRESS P.ACC
21 20 20 57 55 35 21 20 20 20 20 20 20 20 20 0022B P.ACE: .ASCII \!8XL !15AF !5AC !3UB!3(+) -- swa\
20 20 20 20 20 20 74 75 6F 20 20 64 65 70 70 0023A
21 20 20 57 55 35 21 20 20 20 20 20 20 20 20 00244 .ASCII \pped out -- !5UW !AC!/ \
20 20 20 20 20 20 74 75 6F 20 20 64 65 70 70 00253
21 20 20 57 55 35 21 20 20 20 20 20 20 20 20 00262
00 00 49 25 21 20 0026C
010E0056 00274 P.ACD: .ASCII \ !XI\<0><0>
00000000 00278 P.ACD: .LONG 17694806
43 41 35 21 20 46 41 35 31 21 20 4C 58 38 21 0027C P.ACG: .ADDRESS P.ACE
20 20 20 20 29 28 28 33 21 42 55 33 21 20 20 0028B P.ACG: .ASCII \!8XL !15AF !5AC !3UB!3(+) -- swa\
20 20 20 20 20 20 74 75 6F 20 20 64 65 70 70 0029A
21 20 20 57 55 35 21 20 20 20 20 20 20 20 20 002A4 .ASCII \pped out -- !5UW !AC\
00 00 49 25 21 20 002B3
010E0048 002C4 P.ACF: .LONG 17694792
00000000 002C8 P.ACF: .ADDRESS P.ACG
```

```
SD_LNM$SYSTEM= P.ABK
.EXTRN SY$ASCTIM, SYS$TRNLNM
.PSECT $CODE$,NOWRT,2
```

007C 00000 PRINT_DATA:

```
56 00000000G 00 9E 00002 .WORD Save R2,R3,R4,R5,R6 : 0463
55 00000000G 00 9E 00009 MOVAB SHOW$WRITE_LINE, R6
5E A0 AE 9E 00010 MOVAB SY$ASCTIM, R5
52 04 BC 00000040 8F C1 00014 ADDL3 #64, @DATA, SCRATCH : 0508
62 D5 0001D TSTL (SCRATCH) : 0513
01 12 0001F BNEQ 1$
04 00021 RET
58 AE 00 FF676980 8F 00000000G 00 7A 00022 1$: EMUL EXE$GL ABSTIM, #-10000000, #0, TIME : 0521
50 AE 40 8F 9A 00030 MOVZBL #64, DESC : 0522
54 AE 04 BC D0 00035 MOVL @DATA, DESC+4 : 0523
7E D4 0003A CLRL -(SP) : 0527
5C AE 9F 0003C PUSHAB TIME
58 AE 9F 0003F PUSHAB DESC
5C AE 9F 00042 PUSHAB DESC
65 04 FB 00045 CALLS #4, SY$ASCTIM
54 50 D0 00048 MOVL R0, STATUS
03 54 E8 0004B BLBS STATUS, 2$
18 AE 014B 31 0004E BRW 19$ : 0536
1C AE 10 D0 00051 2$: MOVL #16, DESC SYSNODE : 0537
50 AE 20 AE 9E 00055 MOVAB SYSNODEBUF, DESC SYSNODE+4 : 0547
80 00020010 8F D0 0005E MOVAB TRNLNMLST, $$ITMBLKPTR
MOVL #131088, ($$ITMBLKPTR)+
```


	80	20	AE	9E	00065	MOVAB	SYSNODEBUF, (\$\$ITMBLKPTR)+	
	80	18	AE	9E	00069	MOVAB	DESC SYSNODE, (\$\$ITMBLKPTR)+	
		08	80	D4	0006D	CLRL	(\$\$ITMBLKPTR)+	
08	AE		AE	9F	0006F	PUSHAB	TRNLNMLST	0557
		08	01	D0	00072	MOVL	#1, 8(SP)	
		0000'	AE	9F	00076	PUSHAB	8(SP)	
		0000'	CF	9F	00079	PUSHAB	P.ABM	
10	AE	02000000	CF	9F	0007D	PUSHAB	SD_LNM\$SYSTEM	
		10	8F	D0	00081	MOVL	#33554432, 16(SP)	
00000000G	00		AE	9F	00089	PUSHAB	16(SP)	
	54		05	FB	0008C	CALLS	#5, SYS\$TRNLNM	
	05		50	D0	00093	MOVL	R0, STATUS	
		18	54	E8	00096	BLBS	STATUS, 3\$	
			AE	B4	00099	CLRW	DESC_SYSNODE	0559
			27	11	0009C	BRB	8\$	
5F	8F	20	AE	91	0009E	CMPB	SYSNODEBUF, #95	0565
		18	06	12	000A3	BNEQ	4\$	
		1C	AE	B7	000A5	DECW	DESC_SYSNODE	0568
			AE	D6	000A8	INCL	DESC_SYSNODE+4	0569
			53	D4	000AB	CLRL	I	0572
1C	BE	18	3A	3A	000AD	LOCC	#58, DESC_SYSNODE, @DESC_SYSNODE+4	0573
			02	12	000B3	BNEQ	6\$	
			51	D4	000B5	CLRL	R1	
			51	D5	000B7	TSTL	R1	0574
			03	13	000B9	BEQL	7\$	
		18	AE	B7	000BB	DECW	DESC_SYSNODE	0575
			53	D6	000BE	INCL	I	0573
			53	D1	000C0	CMPL	I, #1	
			E8	1B	000C3	BLEQU	5\$	
30	AE	0000'	CF	9E	000C5	MOVAB	P.ABO, ARGLIST	0582
34	AE	18	AE	9E	000CB	MOVAB	DESC SYSNODE, ARGLIST+4	0583
		38	AE	D4	000D0	CLRL	ARGLIST+8	0584
3C	AE	50	AE	9E	000D3	MOVAB	DESC, ARGLIST+12	0585
50	AE		03	C2	000D8	SUBL2	#3, DESC	0586
			7E	D4	000DC	CLRL	-(SP)	0591
		0000'	CF	9F	000DE	PUSHAB	P.ABR	0592
		38	AE	9F	000E2	PUSHAB	ARGLIST	0591
		0000'	CF	9F	000E5	PUSHAB	P.ABP	
			04	FB	000E9	CALLS	#4, SHOW\$WRITE_LINE	
66			62	D5	000EC	TSTL	(SCRATCH)	0603
			01	12	000EE	BNEQ	10\$	
				04	000F0	RET		
51	0C		A2	9E	000F1	MOVAB	12(SCRATCH), R1	0609
			61	D5	000F5	TSTL	(R1)	
			35	15	000F7	BLEQ	13\$	
0E			61	D1	000F9	CMPL	(R1), #14	0610
			30	14	000FC	BGTR	13\$	
02			61	D1	000FE	CMPL	(R1), #2	0613
			20	12	00101	BNEQ	12\$	
		2C	A2	D5	00103	TSTL	44(SCRATCH)	0614
			1B	19	00106	BLSS	12\$	
			12	15	00108	BLEQ	11\$	0617
0E	2C		A2	D1	0010A	CMPL	44(SCRATCH), #14	0618
			0C	14	0010E	BGTR	11\$	
50	2C		A2	D0	00110	MOVL	44(SCRATCH), R0	0619
61	0000'	CF	40	D0	00114	MOVL	RSN_TABLE-4[R0], (R1)	
			17	11	0011A	BRB	14\$	

61	0000'	CF	9E	0011C	11\$:	MOVAB	P.ABT, (R1)	0620			
		10	11	00121		BRB	14\$	0613			
50		61	D0	00123	12\$:	MOVL	(R1), R0	0622			
61	0000'	CF	40	D0	00126	MOVL	STATÉ_TABLE-4[R0], (R1)				
		05	11	0012C		BRB	14\$	0609			
61	0000'	CF	9E	0012E	13\$:	MOVAB	P.ABU, (R1)	0624			
50	24	A2	9E	00133	14\$:	MOVAB	36(SCRATCH), R0	0630			
54		60	D0	00137		MOVL	(R0), STATUS				
	04	A2	D5	0013A		TSTL	4(SCRATCH)	0631			
		07	13	0013D		BEQL	15\$				
60	0000'	CF	9E	0013F		MOVAB	P.ABV, (R0)	0632			
		1B	11	00144		BRB	18\$				
07		60	15	E1	00146	15\$:	BBC	#21, (R0), 16\$	0633		
		60	0000'	CF	9E	0014A	MOVAB	P.ABW, (R0)	0634		
			10	11	0014F		BRB	18\$			
07		60	0E	E1	00151	16\$:	BBC	#14, (R0), 17\$	0635		
		60	0000'	CF	9E	00155	MOVAB	P.ABX, (R0)	0636		
			05	11	0015A		BRB	18\$			
		60	0000'	CF	9E	0015C	17\$:	MOVAB	P.ABY, (R0)	0637	
10	A2	1F	10	A2	C3	00161	18\$:	SUBL3	16(SCRATCH), #31, 16(SCRATCH)	0642	
		04	A2	30	A2	9A	00167	MOVZBL	48(SCRATCH), 4(SCRATCH)	0651	
	08	A2	31	A2	9E	0016C	MOVAB	49(R2), 8(SCRATCH)	0652		
		4F	54	E9	00171	BLBC	STATUS, 22\$	0658			
58	AE	00	FFFE7960	8F	18	A2	7A	00174	EMUL	24(SCRATCH), #-100000, #0, TIME	0661
		50		AE	10	D0	0017F	MOVL	#16, DESC	0662	
		54		AE	40	AE	9E	00183	MOVAB	PROCTIM, DESC+4	0663
				7E	D4	00188	CLRL	-(SP)		0668	
				5C	AE	9F	0018A	PUSHAB	TIME		
				58	AE	9F	0018D	PUSHAB	DESC		
				5C	AE	9F	00190	PUSHAB	DESC		
		65		04	FB	00193	CALLS	#4, SYSSASCTIM			
		54		50	D0	00196	MOVL	R0, STATUS			
		0A		54	E8	00199	BLBS	STATUS, 20\$			
				54	DD	0019C	19\$:	PUSHL	STATUS	0671	
	00000000G	00		01	FB	0019E	CALLS	#1, LIBSSIGNAL			
				04	001A5		RET			0670	
		18	A2	50	AE	9E	001A6	20\$:	MOVAB	DESC, 24(SCRATCH)	0675
			54	E9	001AB		BLBC	STATUS, 22\$		0680	
08	08	BC		04	E1	001AE	BBC	#4, @FLAGS, 21\$		0683	
				52	DD	001B3	PUSHL	SCRATCH		0685	
			0000'	CF	9F	001B5	PUSHAB	P.ABZ		0684	
				1B	11	001B9	BRB	24\$			
				52	DD	001BB	21\$:	PUSHL	SCRATCH	0687	
			0000'	CF	9F	001BD	PUSHAB	P.ACB		0686	
				13	11	001C1	BRB	24\$			
08	08	BC		04	E1	001C3	22\$:	BBC	#4, @FLAGS, 23\$	0691	
				52	DD	001C8	PUSHL	SCRATCH		0693	
			0000'	CF	9F	001CA	PUSHAB	P.ACD		0692	
				06	11	001CE	BRB	24\$			
				52	DD	001DC	23\$:	PUSHL	SCRATCH	0695	
			0000'	CF	9F	001D2	PUSHAB	P.ACF		0694	
		66		02	FB	001D6	24\$:	CALLS	#2, SHOWSWRITE LINE	0700	
		52		40	A2	9E	001D9	MOVAB	64(R2), SCRATCH	0603	
				FF0C	31	001DD	BRW	9\$		0704	
					04	001E0	RET				

; Routine Size: 481 bytes, Routine Base: \$CODE\$ + 01D4

SHOWSYSTEM
V04-000

B 16
16-Sep-1984 01:22:08
14-Sep-1984 12:09:48

VAX-11 Bliss-32 V4.0-742
[CLIUTL.SRC]SHOWSYS.B32;1

Page 22
(7)

SHOWSYSTEM
V04-000

C 16
16-Sep-1984 01:22:08
14-Sep-1984 12:09:48

VAX-11 Bliss-32 V4.0-742
[CLIUTL.SRC]SHOWSYS.B32;1

Page 23
(8)

: 614 0705 1 END
: 615 0706 0 ELUDOM

.EXTRN LIB\$SIGNAL, LIB\$STOP

PSECT SUMMARY

Name	Bytes	Attributes
\$SPLITS	716	NOVEC,NOWRT, RD ,NOEXE,NOSHR, LCL, REL, CON,NOPIC,ALIGN(2)
\$OWNS	112	NOVEC, WRT, RD ,NOEXE,NOSHR, LCL, REL, CON,NOPIC,ALIGN(2)
\$CODES	949	NOVEC,NOWRT, RD , EXE,NOSHR, LCL, REL, CON,NOPIC,ALIGN(2)

Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
_\$255\$DUA28:[SYSLIB]LIB.L32;1	18619	62	0	1000	00:01.9

COMMAND QUALIFIERS

: BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS\$:SHOWSYS/OBJ=OBJ\$:SHOWSYS MSRC\$:SHOWSYS/UPDATE=(ENH\$:SHOWSYS)

: Size: 947 code + 830 data bytes
: Run Time: 00:22.7
: Elapsed Time: 01:14.4
: Lines/CPU Min: 1862
: Lexemes/CPU-Min: 23509
: Memory Used: 204 pages
: Compilation Complete

0057

DIGITAL
CONFIDE